Claims

1. A processor device comprising

an instruction stream transformation unit and

an instruction stream cache

wherein said instruction stream transformation unit transforms code blocks of instructions from an original instruction set architecture to a transformed instruction set architecture.

2. A processor device as in claim 1 further comprising an execute unit

that can directly execute instructions in both said original instruction set architecture and said transformed instruction set architecture.

3. A processor device as in claim 2 wherein said execute unit is an in-order execute unit.

4. A processor device as in claim 2 wherein said execute unit is a dynamic out-of-order execute unit.

5. A processor device as in claim 1 further comprising a working memory connected to said instruction stream transformation unit.

6. A processor device as in claim 1 wherein said instruction stream transformation unit transforms blocks of code which are presumed hyper-blocks.

7. A processor device as in claim 1 wherein said instruction stream transformation unit transforms blocks of code which are denoted as hyper-blocks in original instruction set architecture code.

8. A processor device as in claim 1 wherein said instruction stream cache comprises means of using a tag for each cache line to denote whether the tag is a start of a hyper-block.

9. A processor device as in claim 1 wherein said instruction stream cache comprises means of using a hyper-block ID as an alternative way of addressing a transformed block of code.

10. A processor device as in claim 9 wherein said instruction stream cache comprises means of storing a plurality of hyper-block lines that are chained together using a common hyper-block ID plus pointers.

11. A processor device as in claim 1 wherein said instruction stream cache only hits when a transformed block of code is entered starting from the first instruction of the block.

12. A processor device as in claim 1 wherein said instruction stream transformation unit comprises means of instruction re-ordering.

13. A processor device as in claim 1 wherein said instruction stream transformation unit comprises means of performing predication if-conversion.

14. A processor device as in claim 1 wherein said instruction stream transformation unit comprises means of converting a load instruction into a speculative load and a load activation instruction pair.

15. A processor device as in claim 1 wherein said instruction stream transformation unit comprises a parallel dependency detector circuit.

16. A processor device as in claim 1 wherein said instruction stream transformation unit processes groups of instructions within an instruction window.

17. A processor device as in claim 16 wherein said instruction stream transformation unit processes groups of instructions using overlapping instruction windows.

18. A processor device as in claim 1 wherein said instruction stream transformation unit comprises means of creating a dependency matrix to represent potential dependencies between instructions.

19. A processor device as in claim 18 wherein said instruction stream transformation unit comprises means of creating an operand mapping table to represent potential writes of operands by instructions.

20. A processor device as in claim 1 wherein said instruction stream transformation unit comprises means of performing register renaming.

21. A processor device as in claim 20 wherein said register renaming by the instruction stream transformation unit allocates a set of physical registers that are separate from a second set of registers allocated by dynamic register renaming done by the execute unit.

22. A processor device as in claim 1 wherein the instruction stream transformation unit comprises means of performing an instruction scheduling algorithm.

23. A processor device as in claim 22 wherein said scheduling algorithm is a list scheduling algorithm.

24. A processor device as in claim 1 wherein said execute unit comprises means of performing dynamic memory disambiguation and said transformed instruction set architecture supports notations for indicating ambiguous memory operations.

25. A processor device as in claim 24 wherein said instruction stream transformation unit comprises means of creating an ambiguous memory dependency matrix.

26. A processor device as in claim 24 wherein said instruction stream transformation unit comprises means of converting an ambiguous memory read instruction into a speculative read and a read check instruction pair.

27. A processor device as in claim 1 comprising a dynamically-scheduled execute unit.

28. A processor device as in claim 1 wherein the transformed instruction set architecture comprises dependency notation means of explicitly describing dependencies between instructions.

29. A processor device as in claim 28 wherein said means of explicitly describing dependency notation means allows instructions to be grouped into mini-tuples for dependency notations.

30. A processor device as in claim 28 wherein said dependencies are represented by dependency pointers in the transformed instruction set architecture.

31. A processor device as in claim 28 wherein said dependencies are represented by dependency vectors in the transformed instruction set architecture.

32. A processor device as in claim 1 further comprising means to perform semi-dynamic instruction code re-writing and re-scheduling.

33. A processor device as in claim 1 further comprising at least one run-time history table.

34. A processor device as in claim 33 comprising a value prediction history table, whereby run-time behavior can be recorded for subsequent instruction scheduling.

35. A processor device comprising a predicate history table, whereby run-time behavior can be recorded for subsequent instruction scheduling.

36. A processor device comprising a data hit/miss history table, whereby run-time behavior can be recorded for subsequent instruction scheduling.

37. A processor device comprising an ambiguous memory conflict history table, whereby run-time behavior can be recorded for subsequent instruction scheduling.

38. A method of providing precise interrupts in a processor implementing instruction stream transformation comprising the steps of

mapping an original instruction set architecture instruction to an equivalent group of one or more transformed instruction set architecture instruction(s),

using physical registers that have not been committed to logical registers to hold results, and

allowing the final instruction in said group to commit the physical register result(s) to logical register(s).

39. A method of providing precise interrupts in a processor implementing instruction stream transformation comprising the steps of

assigning an instruction sequence number to each original instruction set architecture instruction starting from the beginning of the code block,

marking each transformed instruction set architecture instruction with the corresponding instruction sequence number, and

committing the results of instructions in order of the instruction sequence numbers.

40. A software method of performing code scheduling comprising the step of building a dependency matrix.

41. A processor device comprising means of executing an instruction set architecture wherein the instruction set architecture can explicitly note dependencies between instructions by using dependency vectors.

42. A processor device comprising means of executing an instruction set architecture wherein the instruction set architecture can explicitly note dependencies between instructions by using dependency pointers.

43. A processor device comprising an instruction stream cache and means of using software routines to perform instruction stream transformation on code blocks.